

## Draft revised ITU-T Recommendation H.264

## Advanced video coding for generic audiovisual services

## Editor's Note:

This document contains (in integrated form)

- Errata report corrections relative to the May 2003 standard (ITU-T Rec. H.264 | ISO/IEC 14496-10) up to and including the disposition of comments to the March 2004 meeting in Munich (starting with JVT-K051r1).
- Fidelity range extensions amendment as of the July 2004 meeting in Redmond in Draft Amendment 1, which was not separately submitted for approval in ITU-T (the non-integrated version is JVT-L047d12 and the integrated version is JVT-L050d5).
- Additional error report corrections reflecting the outcome of the October 2004 meeting in Palma de Mallorca, Spain (with the non-integrated version being JVT-M049d6).
- Additional error report edits reflecting the outcome of the January 2005 meeting in Hong Kong (subclause 7.4.2.11 semantics of `rbsp_stop_one_bit` and `rbsp_alignment_zero_bit` changing "is a single bit equal to" to "shall be equal to", and subclauses 7.3.5.1 and 7.4.5.1 syntax and semantics of `intra_chroma_pred_mode` changing `u(v)` to `ue(v)` and specifying its range of values).

*Author(s) or  
Contact(s):*

Gary Sullivan  
Microsoft Corporation  
One Microsoft Way  
Redmond, WA 98052 USA

Tel: +1 (425) 703-5308  
Fax: +1 (425) 706-7329  
Email: garysull@microsoft.com

Thomas Wiegand  
Fraunhofer Institute for Telecommunications, Heinrich-  
Hertz-Institut  
Einsteinufer 37, D-10587 Berlin, Germany

Tel: +49 (30) 31002-617  
Fax: +49 (30) 392 72 00  
Email: wiegand@hhi.de

Ajay Luthra  
Motorola Corporation  
6420 Sequence Drive  
San Diego, CA 92121 USA

Tel: +1 (858) 404 3470  
Fax: +1 (858) 404 2501  
Email: aluthra@motorola.com

[REFERENCE: JVT-N050d1wcm.doc]

## Title page to be provided by ITU-T | ISO/IEC

DRAFT INTERNATIONAL STANDARD  
 Draft Revised ISO/IEC 14496-10 (E)  
 Draft Revised Rec. H.264 (E)  
 DRAFT ITU-T RECOMMENDATION

## TABLE OF CONTENTS

Foreword.....	xiii
<b>0 Introduction.....</b>	<b>xiv</b>
0.1 Prologue.....	xiv
0.2 Purpose.....	xiv
0.3 Applications.....	xiv
0.4 Publication and versions of this specification.....	xv
0.5 Profiles and levels.....	xv
0.6 Overview of the design characteristics.....	xv
0.6.1 Predictive coding.....	xv
0.6.2 Coding of progressive and interlaced video.....	xv
0.6.3 Picture partitioning into macroblocks and smaller partitions.....	xvi
0.6.4 Spatial redundancy reduction.....	xvi
0.7 How to read this specification.....	xvi
<b>1 Scope.....</b>	<b>1</b>
<b>2 Normative references.....</b>	<b>1</b>
<b>3 Definitions.....</b>	<b>1</b>
<b>4 Abbreviations.....</b>	<b>8</b>
<b>5 Conventions.....</b>	<b>9</b>
5.1 Arithmetic operators.....	9
5.2 Logical operators.....	10
5.3 Relational operators.....	10
5.4 Bit-wise operators.....	10
5.5 Assignment operators.....	10
5.6 Range notation.....	10
5.7 Mathematical functions.....	10
5.8 Variables, syntax elements, and tables.....	11
5.9 Text description of logical operations.....	12
5.10 Processes.....	13
<b>6 Source, coded, decoded and output data formats, scanning processes, and neighbouring relationships.....</b>	<b>13</b>
6.1 Bitstream formats.....	13
6.2 Source, coded, and output picture formats.....	13
6.3 Spatial subdivision of pictures and slices.....	19
6.4 Inverse scanning processes and derivation processes for neighbours.....	19
6.4.1 Inverse macroblock scanning process.....	19
6.4.2 Inverse macroblock partition and sub-macroblock partition scanning process.....	20
6.4.2.1 Inverse macroblock partition scanning process.....	21
6.4.2.2 Inverse sub-macroblock partition scanning process.....	21
6.4.3 Inverse 4x4 luma block scanning process.....	21
6.4.4 Inverse 8x8 luma block scanning process.....	22
6.4.5 Derivation process of the availability for macroblock addresses.....	22
6.4.6 Derivation process for neighbouring macroblock addresses and their availability.....	22
6.4.7 Derivation process for neighbouring macroblock addresses and their availability in MBAFF frames.....	23
6.4.8 Derivation processes for neighbouring macroblocks, blocks, and partitions.....	24
6.4.8.1 Derivation process for neighbouring macroblocks.....	25
6.4.8.2 Derivation process for neighbouring 8x8 luma block.....	25
6.4.8.3 Derivation process for neighbouring 4x4 luma blocks.....	26
6.4.8.4 Derivation process for neighbouring 4x4 chroma blocks.....	26
6.4.8.5 Derivation process for neighbouring partitions.....	26

- Otherwise (ctxIdxOffset is equal to 77), the following applies.
  - The derivation process for neighbouring macroblocks specified in subclause 6.4.8.1 is invoked and the output is assigned to mbAddrA and mbAddrB.
  - Let the variable condTermFlagN (with N being either A or B) be derived as follows.
    - If mbAddrN is available and mb\_type for the macroblock mbAddrN is equal to L\_PCM, condTermFlagN is set equal to 1
    - Otherwise, if any of the following conditions is true, condTermFlagN is set equal to 0
      - mbAddrN is not available or the macroblock mbAddrN is skipped
      - binIdx is equal to 0 and CodedBlockPatternChroma for the macroblock mbAddrN is equal to 0
      - binIdx is equal to 1 and CodedBlockPatternChroma for the macroblock mbAddrN is not equal to 2
    - Otherwise, condTermFlagN is set equal to 1.
  - The variable ctxIdxInc is derived as

$$\text{ctxIdxInc} = \text{condTermFlagA} + 2 * \text{condTermFlagB} + ((\text{binIdx} - 1) ? 4 : 0) \quad (9-5)$$

NOTE – When a macroblock uses an Intra\_16x16 prediction mode, the values of CodedBlockPatternLuma and CodedBlockPatternChroma for the macroblock are derived from mb\_type as specified in Table 7-11.

#### 9.3.3.1.1.5 Derivation process of ctxIdxInc for the syntax element mb\_qp\_delta

Output of this process is ctxIdxInc.

Let prevMbAddr be the macroblock address of the macroblock that precedes the current macroblock in decoding order. When the current macroblock is the first macroblock of a slice, prevMbAddr is marked as not available.

Let the variable ctxIdxInc be derived as follows.

- If any of the following conditions is true, ctxIdxInc is set equal to 0
  - prevMbAddr is not available or the macroblock prevMbAddr is skipped
  - mb\_type of the macroblock prevMbAddr is equal to L\_PCM
  - The macroblock prevMbAddr is not coded in Intra\_16x16 prediction mode and both CodedBlockPatternLuma and CodedBlockPatternChroma for the macroblock prevMbAddr are equal to 0
  - mb\_qp\_delta for the macroblock prevMbAddr is equal to 0
- Otherwise, ctxIdxInc is set equal to 1.

#### 9.3.3.1.1.6 Derivation process of ctxIdxInc for the syntax elements ref\_idx\_10 and ref\_idx\_11

Input to this process is mbPartIdx.

Output of this process is ctxIdxInc.

The interpretation of ref\_idx\_1X and Pred\_LX within this subclause is specified as follows.

- If this process is invoked for the derivation of ref\_idx\_10, ref\_idx\_1X is interpreted as ref\_idx\_10 and Pred\_LX is interpreted as Pred\_L0.
- Otherwise (this process is invoked for the derivation of ref\_idx\_11), ref\_idx\_1X is interpreted as ref\_idx\_11 and Pred\_LX is interpreted as Pred\_L1.

Let currSubMbType be set equal to sub\_mb\_type[ mbPartIdx ].

The derivation process for neighbouring partitions specified in subclause 6.4.8.5 is invoked with mbPartIdx, currSubMbType, and subMbPartIdx = 0 as input and the output is assigned to mbAddrA/mbPartIdxA and mbAddrB/mbPartIdxB.

With ref\_idx\_1X[ mbPartIdxN ] (with N being either A or B) specifying the syntax element for the macroblock mbAddrN, let the variable refIdxZeroFlagN be derived as follows.

- If MbAffFrameFlag is equal to 1, the current macroblock is a frame macroblock, and the macroblock mbAddrN is a field macroblock

$$\text{refIdxZeroFlagN} = ((\text{ref\_idx\_lX}[ \text{mbPartIdxN} ] > 1) ? 0 : 1) \quad (9-6)$$

- Otherwise,

$$\text{refIdxZeroFlagN} = ((\text{ref\_idx\_lX}[ \text{mbPartIdxN} ] > 0) ? 0 : 1) \quad (9-7)$$

Let the variable `predModeEqualFlag` be specified as follows.

- If the macroblock `mbAddrN` has `mb_type` equal to `P_8x8` or `B_8x8`, the following applies.
  - If `SubMbPredMode(sub_mb_type[ mbPartIdxN ])` is not equal to `Pred_LX` and not equal to `BiPred`, `predModeEqualFlag` is set equal to 0, where `sub_mb_type` specifies the syntax element for the macroblock `mbAddrN`.
  - Otherwise, `predModeEqualFlag` is set equal to 1.
- Otherwise, the following applies.
  - If `MbPartPredMode(mb_type, mbPartIdxN)` is not equal to `Pred_LX` and not equal to `BiPred`, `predModeEqualFlag` is set equal to 0, where `mb_type` specifies the syntax element for the macroblock `mbAddrN`.
  - Otherwise, `predModeEqualFlag` is set equal to 1.

Let the variable `condTermFlagN` (with `N` being either `A` or `B`) be derived as follows.

- If any of the following conditions is true, `condTermFlagN` is set equal to 0
  - `mbAddrN` is not available
  - the macroblock `mbAddrN` has `mb_type` equal to `P_Skip` or `B_Skip`
  - The macroblock `mbAddrN` is coded in Intra prediction mode
  - `predModeEqualFlag` is equal to 0
  - `refIdxZeroFlagN` is equal to 1
- Otherwise, `condTermFlagN` is set equal to 1.

The variable `ctxIdxInc` is derived as

$$\text{ctxIdxInc} = \text{condTermFlagA} + 2 * \text{condTermFlagB} \quad (9-8)$$

### 9.3.3.1.1.7 Derivation process of `ctxIdxInc` for the syntax elements `mvd_l0` and `mvd_l1`

Inputs to this process are `mbPartIdx`, `subMbPartIdx`, and `ctxIdxOffset`.

Output of this process is `ctxIdxInc`.

The interpretation of `mvd_lX` and `Pred_LX` within this subclause is specified as follows.

- If this process is invoked for the derivation of `mvd_l0`, `mvd_lX` is interpreted as `mvd_l0` and `Pred_LX` is interpreted as `Pred_L0`.
- Otherwise (this process is invoked for the derivation of `mvd_l1`), `mvd_lX` is interpreted as `mvd_l1` and `Pred_LX` is interpreted as `Pred_L1`.

Let `curSubMbType` be set equal to `sub_mb_type[ mbPartIdx ]`.

The derivation process for neighbouring partitions specified in subclause 6.4.8.5 is invoked with `mbPartIdx`, `curSubMbType`, and `subMbPartIdx` as input and the output is assigned to `mbAddrA/mbPartIdxA/subMbPartIdxA` and `mbAddrB/mbPartIdxB/subMbPartIdxB`.

Let the variable `complex` be derived as follows.

- If `ctxIdxOffset` is equal to 40, `complex` is set equal to 0.
- Otherwise (`ctxIdxOffset` is equal to 47), `complex` is set equal to 1.

Let the variable `predModeEqualFlag` be specified as follows.

- If the macroblock `mbAddrN` has `mb_type` equal to `P_8x8` or `B_8x8`, the following applies.

- If  $\text{SubMbPredMode}(\text{sub\_mb\_type}[\text{mbPartIdxN}])$  is not equal to  $\text{Pred\_LX}$  and not equal to  $\text{BiPred}$ ,  $\text{predModeEqualFlag}$  is set equal to 0, where  $\text{sub\_mb\_type}$  specifies the syntax element for the macroblock  $\text{mbAddrN}$ .
- Otherwise,  $\text{predModeEqualFlag}$  is set equal to 1.
- Otherwise, the following applies.
  - If  $\text{MbPartPredMode}(\text{mb\_type}, \text{mbPartIdxN})$  is not equal to  $\text{Pred\_LX}$  and not equal to  $\text{BiPred}$ ,  $\text{predModeEqualFlag}$  is set equal to 0, where  $\text{mb\_type}$  specifies the syntax element for the macroblock  $\text{mbAddrN}$ .
  - Otherwise,  $\text{predModeEqualFlag}$  is set equal to 1.

Let the variable  $\text{absMvdCompN}$  (with N being either A or B) be derived as follows.

- If any of the following conditions is true,  $\text{absMvdCompN}$  is set equal to 0
  - $\text{mbAddrN}$  is not available
  - the macroblock  $\text{mbAddrN}$  has  $\text{mb\_type}$  equal to  $\text{P\_Skip}$  or  $\text{B\_Skip}$
  - The macroblock  $\text{mbAddrN}$  is coded in Intra prediction mode
  - $\text{predModeEqualFlag}$  is equal to 0
- Otherwise, the following applies
  - If  $\text{compIdx}$  is equal to 1,  $\text{MbaffFrameFlag}$  is equal to 1, the current macroblock is a frame macroblock, and the macroblock  $\text{mbAddrN}$  is a field macroblock
 
$$\text{absMvdCompN} = \text{Abs}(\text{mvd\_LX}[\text{mbPartIdxN}][\text{subMbPartIdxN}][\text{compIdx}]) * 2 \quad (9-9)$$
  - Otherwise, if  $\text{compIdx}$  is equal to 1,  $\text{MbaffFrameFlag}$  is equal to 1, the current macroblock is a field macroblock, and the macroblock  $\text{mbAddrN}$  is a frame macroblock
 
$$\text{absMvdCompN} = \text{Abs}(\text{mvd\_LX}[\text{mbPartIdxN}][\text{subMbPartIdxN}][\text{compIdx}]) / 2 \quad (9-10)$$
  - Otherwise,
 
$$\text{absMvdCompN} = \text{Abs}(\text{mvd\_LX}[\text{mbPartIdxN}][\text{subMbPartIdxN}][\text{compIdx}]) \quad (9-11)$$

The variable  $\text{ctxIdxInc}$  is derived as follows

- If  $(\text{absMvdCompA} + \text{absMvdCompB})$  is less than 3,  $\text{ctxIdxInc}$  is set equal to 0.
- Otherwise, if  $(\text{absMvdCompA} + \text{absMvdCompB})$  is greater than 32,  $\text{ctxIdxInc}$  is set equal to 2.
- Otherwise  $(\text{absMvdCompA} + \text{absMvdCompB})$  is in the range of 3 to 32, inclusive,  $\text{ctxIdxInc}$  is set equal to 1.

**9.3.3.1.1.8 Derivation process of  $\text{ctxIdxInc}$  for the syntax element  $\text{intra\_chroma\_pred\_mode}$**   
Output of this process is  $\text{ctxIdxInc}$ .

The derivation process for neighbouring macroblocks specified in subclause 6.4.8.1 is invoked and the output is assigned to  $\text{mbAddrA}$  and  $\text{mbAddrB}$ .

Let the variable  $\text{condTermFlagN}$  (with N being replaced by either A or B) be derived as follows.

- If any of the following conditions is true,  $\text{condTermFlagN}$  is set equal to 0
  - $\text{mbAddrN}$  is not available
  - The macroblock  $\text{mbAddrN}$  is coded in Inter prediction mode
  - $\text{mb\_type}$  for the macroblock  $\text{mbAddrN}$  is equal to  $\text{L\_PCM}$
  - $\text{intra\_chroma\_pred\_mode}$  for the macroblock  $\text{mbAddrN}$  is equal to 0
- Otherwise,  $\text{condTermFlagN}$  is set equal to 1.

The variable  $\text{ctxIdxInc}$  is derived by

$$\text{ctxIdxInc} = \text{condTermFlagA} + \text{condTermFlagB} \quad (9-12)$$